ABSTRACT

Computing paradigm has perceived a logical shift from CPU towards application specific GPU, FPGA, CPLD due to slow down of Moore's Law, operating system overheads, serial data processing, memory management, power efficiency and speed. The performance increase of general-purpose CPUs & GPUs is unable to match with the advances in peripheral interfaces, reconfigurable logic deployed in FPGAs provides several exceptional properties that may be able to deliver desired performance. FPGA based digital circuits provide an intermediate arrangement between ASIC and CPU with regards to through-put, latency, portability and design time.

Random numbers are mostly used in securing information in several applications from electronic commerce to authentication. Moreover, they are used in applications such as games, lotteries, statistical data analysis, sampling, simulations and testing machine learning algorithms. Technologies generating true or pseudo random numbers needs to be developed in areas such as secure communications, security systems and online banking. Currently true random number generators are expensive, low bandwidth and speed, non-compatible with FPGA or heterogeneous architectures. Therefore, design and development of alternative and affordable true random-number generators is focused by several researchers. National Institute of Standards and Technology (NIST), provides a framework for examination of the statistical properties of generated bit streams to determine whether those bit streams fall within the bounds of statistical randomness.

In this research, challenges associated with FPGA implementation of pseudo random number generator for image encryption algorithms will be addressed. Firstly, to comprehensively study the feasibility and suitability of FPGA implementation of random number generator algorithms in comparison with conventional CPU / GPU / AISC. Secondly to study practicability of commonly deployed pseudo random number generator algorithms with respect FPGA implementation to deliver desired throughput. Finally, to explore different architectures for digital design and development, timing simulation, synthesis and FPGA implementation of machine learning resistant pseudo random number generator for image encryption algorithm and its performance analysis using important metrices such as speed, power and resource utilization.

CONTENTS

Sr. No.		Contents	Page No.
i		Abstract	iii
ii		List of Tables	iv
iii		List of Figures	v
iv		Abbreviations	vi
1		Title	1
2		Aim of Study	1
3		Objectives of Study	1
4		Background of Study	1
	4.1	Introduction	1
	4.2	Random Number Generator	2
	4.3	Literature Survey	3
	4.4	Applications	8
5		National Status	8
6		International Status	9
7		Study Hypothesis	9
8		Study Rationale	11
9		Research Methodology	12
10		Statistical Plan	13
11		Limitations of Study	14
12		Expected Outcomes of Study	14
		Bibliography	15

LIST OF FIGURES

Figure No.	Title	Page No.
1	True random generator conventional system	3
2	Ring oscillator-based TRNG	4
3	DCM metastability generator	4
4	Enhanced chaotic PRNG	5
5	Matlab HDL coder workflow	5
6	TRNG BFD architecture	6
7	DCM based TRNG core and the adopted testing architecture	7
8	ML resistant PRNG	8
9	Proposed block diagram	10
10	Research Methodology	11

LIST OF TABLE

Table No.	Title	Page No.
1	Timeline	14

1. Title: FPGA Based Machine Learning-Resistant Pseudo-Random Number Generator for Image Encryption

2. Aim of Study:

The aim of research is FPGA based design, development, timing simulation, synthesis and implementation of "Machine Learning-Resistant pseudo-random number generator for image encryption" and its performance analysis using important matrices such as NIST test, speed, power and resource utilization.

3. Objectives of Study

1. Design, development, timing simulation, synthesis and FPGA implementation of pseudo random number generator.

2. Performance analysis using NIST test suite.

3. Explore architecture with machine learning resistant characteristics for pseudo random number generator.

4. Demonstrate use of generated random numbers for image encryption algorithm through performance parameters.

4. Background of Study

4.1 Introduction

Internet of things (IoT) will connect several billions of devices and generate 100 terabytes of data per year. This tremendous amount of data is generated due to explosive growth in internet and digital technology devices such as mobile phones, smart watches, tablets, laptops, smart electrical vehicles, surveillance cameras, drones, production monitoring, environment monitoring and many more. Large quantity of data is available that can be collected, analysed, processed and comprehended. It has been possible due to advancement in the VLSI technology commencing from the transistor to application specific integrated circuits (ASIC) and digital logic gates to computing integrated circuits such as CPU, GPU, FPGA etc. Computing paradigm has perceived a logical shift from CPU towards application specific GPU, FPGA, CPLD due to slow down of Moore's Law, operating system overheads, serial data processing, memory management, power efficiency and speed. Pseudo random number generators (PRNGs) deployed using software systems results in low quality and throughput. Additionally high processing time is also observed owing to highly correlated

data and serial execution of programs [1] [2] [3]. FPGA based digital circuits provide an intermediate arrangement between ASIC and CPU with regards to through-put, latency, portability and design time. FPGA enables arbitrary different digital circuits such as on-chip memory, combinatorial logic blocks, digital signal processor units and logic gates to be configured through interconnect between them. It enables to deploy several applications from communication interfaces, cryptography to complex analytical engines. This system integration on FPGA allows data flow process modelling taking complete advantage of pipeline and parallelism features. By using a considerably larger percentage of the circuitry in FPGAs for actual operations in a dataflow fashion, data moves between successive compute steps can be minimized. While ASIC designs have traditionally been tested and prototyped on FPGAs, new designs from many vendors are specifically aimed at data centres and cloud infrastructures [4] [5] [6]. FPGAs are a desirable and important component of many major cloud providers' data centre strategy because they offer better opportunities for increased performance and energy efficiency for the reasons like, Custom datatypes, Custom memory hierarchies and connectivity, No instructions, Very deep pipelining, Abundant parallelism, Low power consumption.

4.2Random Number Generator

A physical entropy source that uses a physical noise source that demonstrate variations in its all or some properties or characteristics is a source of random number. A small deviation in the properties of the source at a certain time gradually effects future conditions in the system. Such a random noise source can be deployed in order to generate an infinite sequence of bits, by converting the analogue values of the numbers into digital form using several techniques. If true randomness does not really exist or becomes difficult to develop or generate, then the design of unpredictable pseudorandom number generators is required. It uses random processing through which sources of uncertainty such as noise are extracted, amplified and digitized. Field-programmable gate arrays (FPGAs) appear to be useful for implementing true random number generators (TRNGs) owing to their internal architecture which provides complete support to bitwise operations [7] [8] [9]. In FPGAs, metastability and jitter provide the essential randomization. Additionally, it is preferable to design a TRNG that is independent of the technology used to build the FPGA and allows portability between devices [10]. TRNG design in FPGAs is more challenging because it must meet low power and area, high speed and throughput requirements without comprising the statistical quality of the desired results for intended applications.

4.3Literature Survey

A silicon PN diode, amplifier, pulse shaping circuit, discriminating circuit, calculating circuit, random number circuit, and controller circuit make up the conventional true random number production system (see figure 1). The source generating circuit, a silicon PN diode, produces a signal that is entirely dependent on particles that have been identified. Preamplifiers work by amplifying the detection signal to produce an amplification signal and increasing the detected signal's magnitude. The counter circuit's job is to count the number of clock seconds that elapse between two successive pulses in the pulse signal. Based on the outcome of the conversion, the random-number output circuit can verify the binary count values produced by the counter.

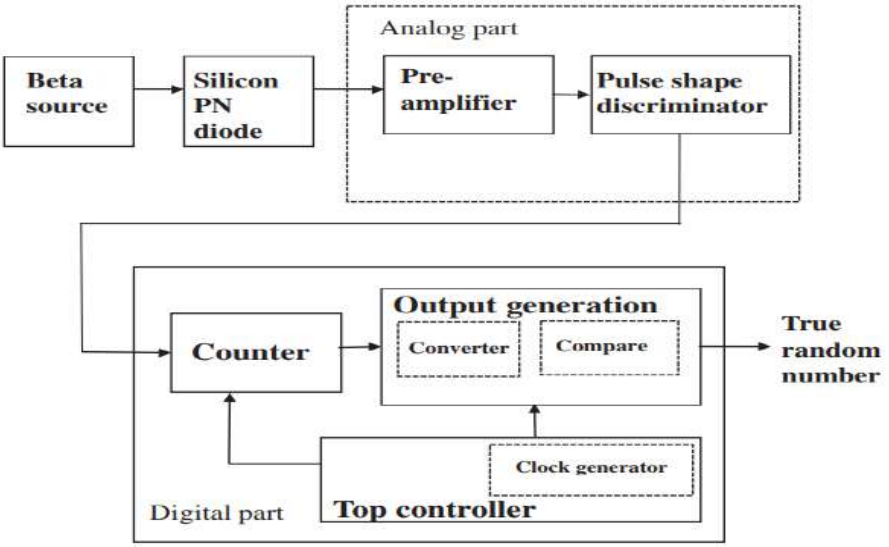


Fig. 1. True random generator conventional system

A lot of scholars have been studying TRNGs lately. Device shot and thermal noises are random sources that are utilized to produce genuinely random numbers. The working of ring oscillator is demonstrated in Fig. 2 [19]. A “D flip-flop” is used by a low-frequency oscillator to sample the high-frequency oscillator's output. An oscillator transforms the device's thermal noise into jitter, which is then employed as a random source [27] [28] [29] [30]. The output bit streams will be unpredictable when each oscillator wanders in frequency at random intervals throughout each cycle [24] [29]. The achievable jitter and the oscillators' mean frequency spacing have a significant impact on the degree of randomization. Better randomness is achieved with greater jitter and frequency separation [10].

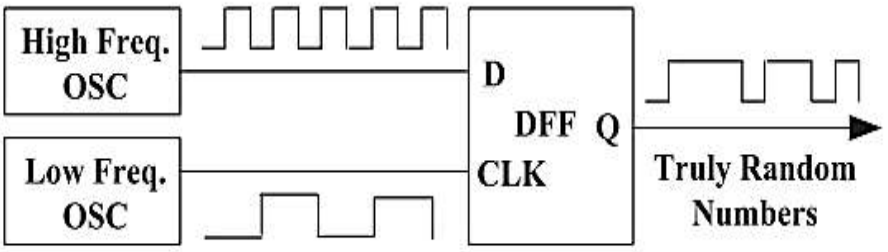


Fig. 2. Ring oscillator-based TRNG

It has been shown that using clock managers to construct TRNGs on FPGA devices with more flexibility is an efficient technique to achieve this. With contemporary Xilinx FPGAs, an additional flexibility known as hardware primitives is made possible by the Digital Clock Manager (DCM), which seeks to produce one or more output clock signals with programmable frequencies [1]. DCMs use their Dynamic Partial Reconfiguration Port (DRP) to dynamically configure their parameters. As a result, it causes the clock signal and flip flop input signal's frequency to be adjusted (see fig. 3).

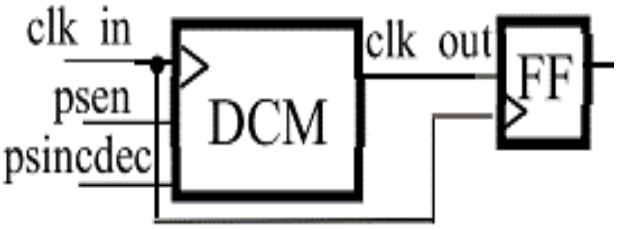


Fig. 3. DCM metastability generator

A chaotic PRNG [11] [12] [13] [14] refer fig. 4 can be designed with the use of digitized chaotic map using computational algorithms. A seed is composed of initial value and state variable, that is responsible for generation of sequence of elements. It is possible that there may be correlations among the generated bits within a number or series of numbers generated. Since improving statistical qualities is crucial, only a small portion of each x_i typically the least significant bits (LSBs) because of their low correlation are chosen to create the random sequences [13].

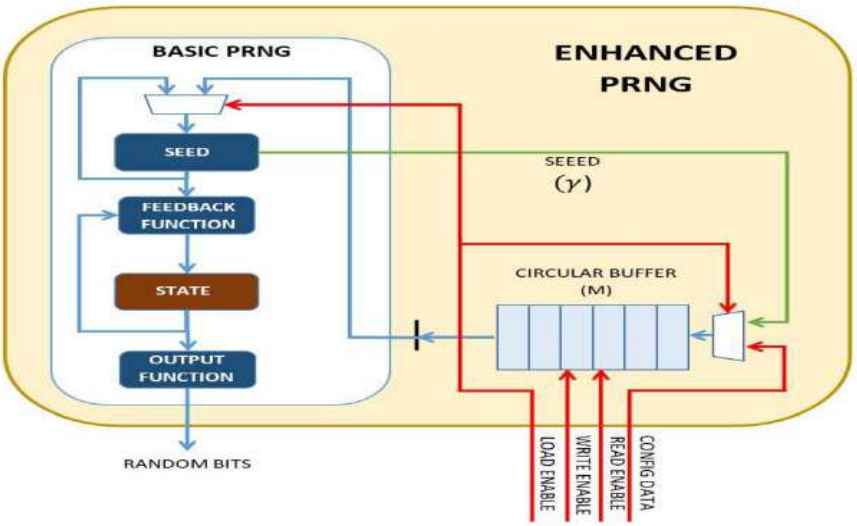


Fig. 4. Enhanced chaotic PRNG

When a chaotic map is digital and its word length is n bits, it can have a maximum of n alternate values for each value. Moreover, the value of one element finds the value of the following element for a given initial value. Therefore, the sequence will repeat itself after a maximum of 2n repetitions. Most of the NIST randomness tests are not passed by these short-period sequences [37].

With MathWorks 2023's MATLAB HDL Coder, FPGA design is possible. Figure 5 illustrates its workflow, which has two input methods: Simulink and MATLAB script. Understanding the I/O types is necessary for the MATLAB script to generate the HDL code. This is required for the pins on the FPGA. Additionally, all types must be made interoperable with one another. To enable the mapping of all functions onto hardware, it also needs to identify all libraries and functions that are used [36].

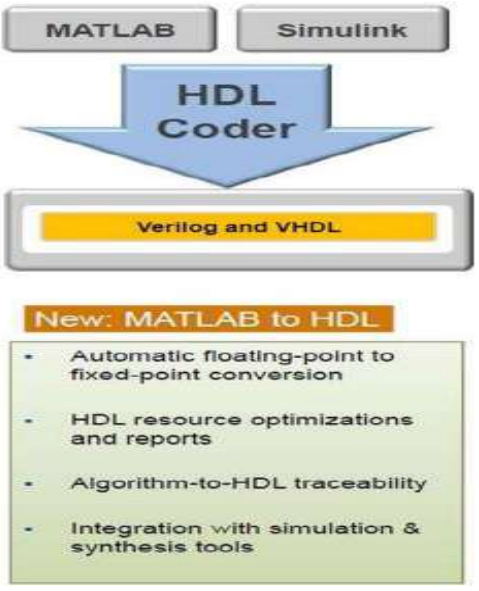


Fig. 5. Matlab HDL coder workflow

Pseudorandom number generators (PRNGs), which use a deterministic process to extend a brief random string into a set of random looking numbers, are adequate for many purposes [15] [16] [17]. For cryptographic applications, however, it is critical to create pseudo random bits that are unpredictably recognised even by the most powerful attacker. Furthermore, pseudo random number generator with quality output which is tolerance to attacks can be built, which is called TRNG [34]. PRNGs uses a known algorithm to generate a huge number, which are completely random from a small quantity of unpredictability numbers called seed. PRNGs are divided into two parts cryptographic and non-cryptographic. The linear congruential generators present in many programming libraries are noncryptographic PRNGs. These libraries statistically acceptable random numbers, but they can't use for cryptographic keying applications. This PRNGs takes input as seed and gives output. The seed used for PRNG is mathematically derived so we can derive the internal state and output of PRNG easily. Attacker can easily predict the outputs if he knew previous outputs so there is security problems associated with PRNG [19] [21] [23]. A nondeterministic irregular bit generator employment a nondeterministic source to produce randomness. Most of the randomness (random seed) produced by unpredictable natural process such as atmospheric noise, jitter noise and nuclear decay. The major performance of TRNG depends upon the quality of seed produced by entropy source. We can create randomness from within deterministic system. Computers which don't have hardware entropy source try to obtain seed from embedded devices of hardware like keyboard and hard drives, memory, etc. The entropy derived from the sources like keyboard is not at all sufficient, so it is better to have hardware entropy source. Computationally bounded test that can assess a RNG's output and authoritatively affirm that the output is random, since no test is perfect to detect randomness so defects are frequently overlooked. Entropy source should produce output regularly with some frequency. However, some software system stores a seed value in hardware which have security problems, since there are some problems with entropy source. Hardware that provides a well-designed, efficient, and easy-to-use hardware entropy source is the greatest answer to these problems [5] [23].

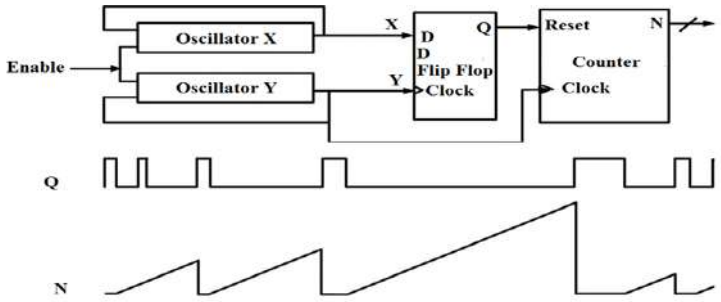


Fig. 6. TRNG BFD architecture

A new way was given to quickly create TRNGs on FPGA devices by utilizing the hardware primitives of Digital Clock Manager (DCM) [31] to alter the phase shift between two clock signals during runtime. In order to push one or more flip-flops (FFs) into the metastability region—which is employed as a source of randomness—the self-tuning approach that is being presented automatically modifies the phase difference of two clock signals. To further boost the randomness of the generated bits, a novel application of fast transport chain hardware primitives is suggested [32]. The paper concludes with a description of an efficient on-chip post-processing approach that maintains TRNG throughput. The Xilinx Zynq XC7Z020 System on Chip (SoC) is used to implement the suggested TRNG design. The XOR gate, which completes the DCM-based architecture [1].

By using a map of this type, a sequence of elements $\{x_i\}$ is created, beginning with a seed made up of $x_0$and γ. Each element can be used to create a binary random generator since it is represented by a set number of bits. Nonetheless, correlations may exist between the bits that make up an element x_i. Therefore, only a few bits of each x_i are normally used to generate the random sequences, usually the least significant bits (LSBs) because they offer a low correlation, in order to gain superior statistical features [33]. By employing only, a little amount of additional resources above the 32-bit logistic map generator, the system has demonstrated its ability to produce sequences with strong statistical qualities by passing the NIST randomness tests [18] [26]

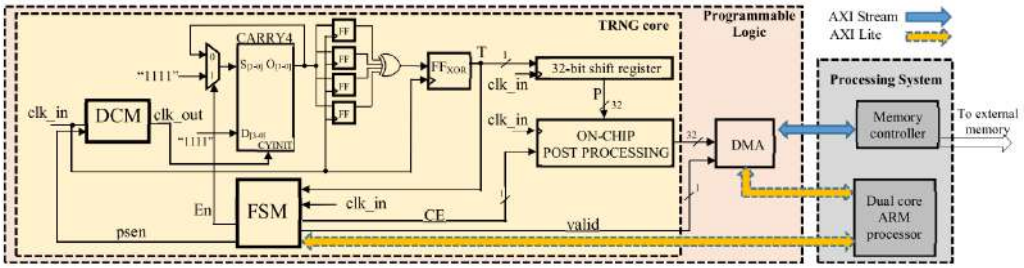


Fig. 7. DCM based TRNG core and the adopted testing architecture

Traditional pseudo-random number generators (PRNGs) are exposed to “Machine Learning (ML) attacks. ML attacks" can also be used to breach hardware security primitives such as "Physical Unclonable Functions (PUF)". But the primary security distinction between PUFs and conventional PRNGs is that a typical PRNG can be broken simply by training its output data. A new architecture composed of a PUF and two regular PRNGs is proposed: PRNG1 and PRNG2 [7]. First, the output of PRNG1 is fed into R1 “PUF” to generate an intermediate data R2. The intermediate data R2 is added to the R3 output of the PRNG2 to produce the

final R4 output of the new PRNG. Since the input challenge of the PUF in Figure 8 is hidden in the R1 chip, the adversary cannot model the secret information of the PUF even using ML techniques [35]. In the new PRNG, the PUF input challenge was masked so that the adversary could not model all elements. When a "long short-term memory (LSTM) attack" [22] When carried out on the suggested PRNG, the training precision is approximately 52%. In contrast, if the "LSTM attack" is implemented on a normal PRNG, its training accuracy will be above 99% [7]. A summary of recent techniques proposed in the international literature for PRNG on FPGAs is presented in Table 1 & 2.

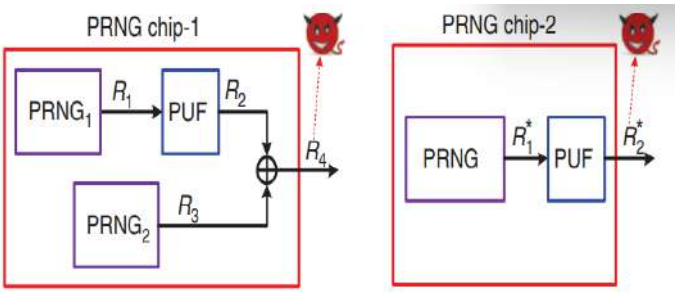


Fig. 8. ML resistant PRNG

To speed up "Encryption and Decryption", the BFV system suggested a thorough creating and putting into practice a hardware architecture [25] [32]. Additionally, we generate Gaussian noise in cryptographic procedures using an integrated Gaussian pseudorandom number generator (PRNG). According to experimental results, the accelerator runs 1024–22 times faster than alternative techniques for the same polynomial degree. Moreover, the suggested Gaussian PRNG has a correlation that is twice as good as the rotation-only PRNG. Lastly, on an "Intel Core i7 processor" with 8GB of RAM, our accelerator offers speedups of up to 6.8x overall, as well as up to 9x for "Encryption" and 3.5x for "Decryption" when compared to "Microsoft SEAL."

Applications

In the fields of cryptography, communications, industrial equipment and measurements, Monte Carlo simulations, random filling, mobile and computer games, laboratory testing, and the use of evolutionary algorithms to characterize networks, random number generators are employed to find the best input random variables.

5. National Status:

FPGA-based machine learning-resistant pseudo-random number generators (PRNGs) for image encryption are gaining momentum India due to growing interest, government and

defence support, academic-industry collaboration, and technological innovation. Recently a comparative study of LFSR based PRNGs with different primitive feedback polynomials was presented. Different architectures are explored for design of PRNGs. FPGA prototype of low latency BBS PRNG using FPGA chip Virtex5 XC5VLX110T was presented. BBS is proved cryptographically secure for PRNGs with f = 92MHz and delay 2.8 ms. Several academicians have shown interest in FPGA prototyping and ASIC design of PRNGs. However, due to tremendous development in AI, design of machine learning resistant architecture for PRNGs is very important.

6.International Status:

FPGA-based machine learning-resistant pseudo-random number generators (PRNGs) for image encryption are gaining attention internationally due to the growing threat of ML-based attacks on traditional encryption methods. Countries like the U.S., China, and Europe are exploring these solutions, with academic and industry collaborations in sectors like defence, healthcare, and finance. International collaborations and open-source contributions are also being pursued to accelerate the development and adoption of these technologies. Several researchers and engineering practitioners have deployed PRNGs for deferent applications. Hardware implementation of digital pseudo-random number generators for real-time applications was presented using FPGA (Xilinx Zynq-7000) with operating frequency at 192.446Mhz and 32-bit, maximum throughput at 73,899 Mbps. Resource constraints authentication mechanisms challenges include lightweight cryptography and secure storage. Recently machine learning-resistant pseudo-random number generator are explored for several applications.

7.Study Hypothesis

It is proposed to explore metastability features available in FPGA to digitally design a

machine learning resistant pseudo random number generator for image encryption applications.

Proposed Block Diagram:

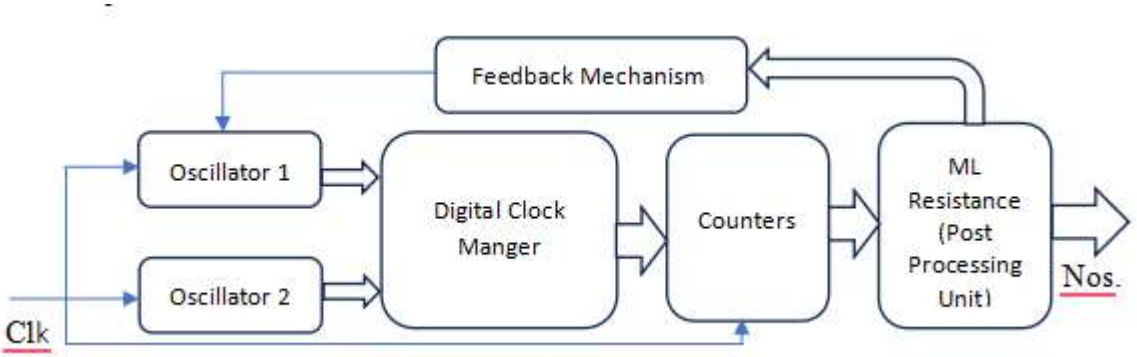


Fig. 9. Proposed block diagram

Proposed block diagram may consist of two ring oscillators, digital clock manager, asynchronous counters and post processing unit modified for machine learning resistant characteristics. A feedback module may be included in the design to provide additional metastability to the circuit by providing random feedback signal. The feedback signal may alter the frequency of the oscillators that may generate random counts. The random numbers generated will be tested to satisfy NIST test suite.

- Architectures deployed for implementation of PRNGs are as follows

  a.Metastability generator through

  i.Linear feedback shift registers

  ii.Asynchronous counters

  iii.Digital clock manager

  b.Chaotic systems

  c.Post processing technique are used for

  i.Enhancement for randomness properties

  ii.Machine learning resistance

- Software: System Verilog / VHDL

  a.Xilinx ISE WebPACK Design Software

  b.AMD Vivado Design Suite

- Hardware: Xilinx / AMD FPGA Kit

  a.PYNQ-Z2 board featuring the ZYNQ XC7Z020-1CLG400C SoC

8. Study Rationale

The important observations identified after detailed literature review in FPGA implementation of PRNGs is depicted as follows

- Computing paradigm has perceived a logical shift from CPU towards application specific GPU, FPGA, CPLD due to slow down of Moore's Law, operating system overheads, serial data processing, memory management, power efficiency and speed.
- The increase in performance parameters of general-purpose CPUs & GPUs is unable to match with the newly adopted peripheral interfaces.
- FPGA based digital circuits provide an intermediate arrangement between ASIC and CPU with regards to through-put, latency, portability and design time.
- FPGA implementation forms the basis of ASIC implementation.
- True random number generators are expensive, low bandwidth and speed, non-compatible with FPGA or heterogenous architectures.
- Ring oscillators that are used with inverters are generally suitable for low power and area allocations, and they can withstand noise levels of 1/f.
- Asynchronous modulo counter can be successfully replaced with ring oscillators and PLL to explore device independent operation.
- LUT-SR TRNGs do not produce as high-quality random bitstreams as LUT-FIFO TRNGs do, LUT-FIFO TRNGs leverage RAM blocks found in FPGA to produce high-quality random bitstreams
- Although Altera FPGAs do not have DPR or DCM facilities, Xilinx does, which suggests unambiguous importability.
- Metastability is improved by the feedback mechanism derived from the generated random number. Post processing scheme may be deployed to enhance the statistical characteristics of the generated random numbers.
- Hyperchaotic systems have more complex behaviours and better random-like characteristics than general chaotic systems.
- Chaotic systems are computationally complex and needs more hardware resources.
- Conventional pseudo-random number generator (PRNG) is vulnerable to machine

learning (ML) attacks since algorithms are used to generate the random number.

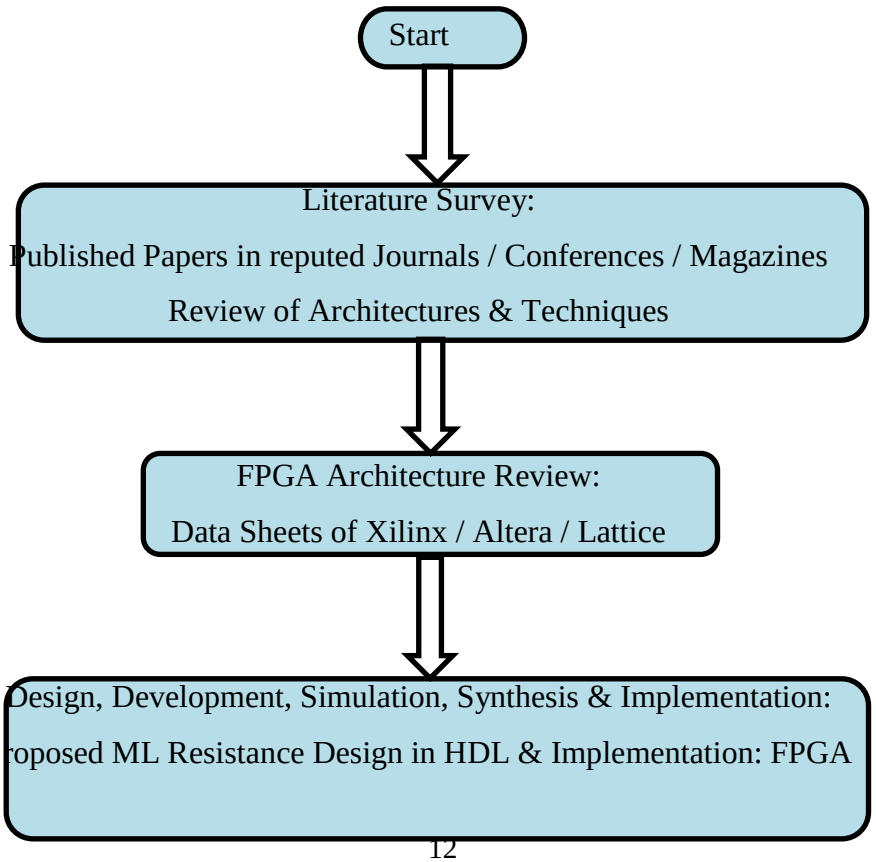
- Chaos-based TRNGs and PRNGs may not be used in limited power applications because it has been observed that the increased complexity leads to reduced throughput and higher power dissipation.
- FPGA implementation of random number generator for accelerating edge computing algorithms for big data analytics is necessary.

Research Gaps:

- FPGA based digital circuits provide an intermediate arrangement between ASIC and CPU with regards to through-put, latency, portability and design time.
- True random number generators are expensive, low bandwidth and speed, non-compatible with FPGA or heterogenous architectures.
- Post processing scheme may be deployed to enhance the statistical characteristics of the generated random numbers especially machine learning resistant.
- FPGA implementation of random number generator for accelerating encryption algorithms for security is necessary.

9. Research Methodology

The detailed research methodology is explained using flow chart in figure 10.


Start
Literature Survey:
Published Papers in reputed Journals / Conferences / Magazines
Review of Architectures & Techniques
FPGA Architecture Review:
Data Sheets of Xilinx / Altera / Lattice
Design, Development, Simulation, Synthesis & Implementation:
roposed ML Resistance Design in HDL & Implementation: FPGA

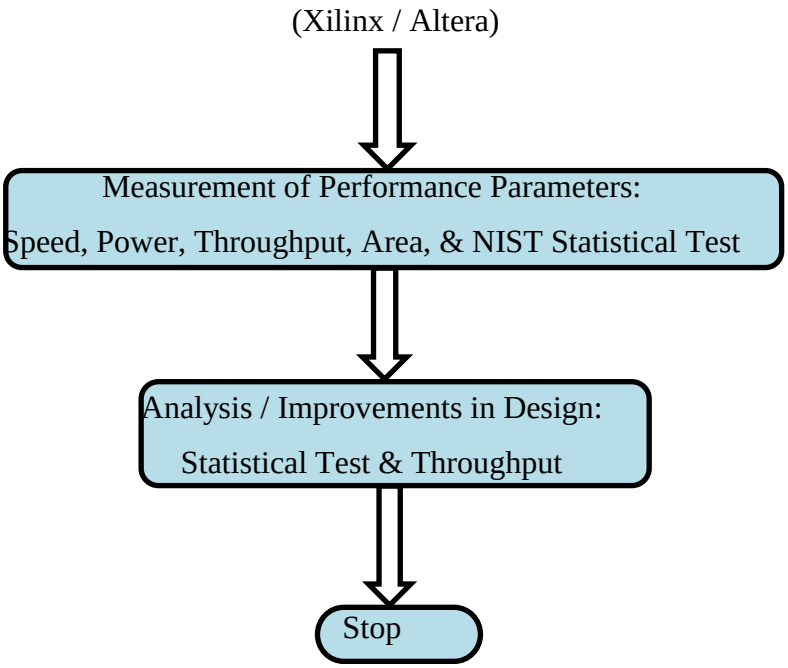

(Xilinx / Altera)
Measurement of Performance Parameters:
Speed, Power, Throughput, Area, & NIST Statistical Test
Analysis / Improvements in Design:
Statistical Test & Throughput
Stop

Fig. 10. Research Methodology

10. Statistical Plan

The detailed statistical plan for the proposed research work is elaborated in table 3.

Table 1. Timeline

Particulars	Duration (months)					
	0 – 6	7 – 12	13 – 18	19 – 24	25 – 30	31 - 36
Literature Survey						
FPGA Review & Selection						
Design, Simulation & Implementation of PRNG						
Performance Analysis & Improvements in design						
Publication of research work in reputed conferences / journals						
Thesis Writing						

11.Limitations of Study

Computing paradigm has perceived a logical shift from CPU towards application specific GPU, FPGA, CPLD due to slow down of Moore's Law, operating system overheads, serial data processing, memory management, power efficiency and speed.

It is proposed to explore metastability in the implementation of digital design in FPGA as a source of randomness. Hence, the scope of the proposed research is limited to use of FPGAs and its architecture that supports metastability for implementation of pseudo random number generator. Further it is desired to explore the generated random numbers for image encryption application.

12.Expected Outcomes of Study

A comprehensive digital design and its FPGA implementation of machine learning resistant pseudo random number generator (PRNG) that satisfies NIST test and especially applied for image encryption scheme is desired.

The PRNG will be successfully designed and developed, with timing simulations and synthesis verifying the design, correctness and performance.

FPGA implementation may demonstrate that the system operates efficiently with minimal hardware resources and achieves the desired randomness properties.

The PRNG needs to pass the NIST test suite, confirming that it generates statistically random numbers that meet industry standards. This analysis highlights the PRNGs robustness and suitability for secure applications.

The study will identify key architectural features that enhance resistance to machine learning-based attacks. These features include unpredictable number sequences and enhanced entropy sources, improving the overall security of the PRNG.

The PRNGs random numbers were successfully used in an image encryption algorithm, showcasing secure and efficient encryption.

Bibliography

[1]	Fabio Frustaci, Fanny Spagnolo, Stefania Perri and Pasquale Corsonello, “A High-Speed FPGA-Based True Random Number Generator Using Metastability with Clock Managers,” IEEE Transactions on Circuits and Systems-II: Express Briefs, Vol. 70, No. 2, pp. 756 – 760, 2023.
[2]	S. Akter, K. Khalil and M. Bayoumi, "Efficient Pseudo Random Number Generator (PRNG) Design on FPGA," 2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS), Richardson, TX, USA, 2024, pp. 1-5.
[3]	F. Zhang, J. Tang, Z. Zhang, Z. Huang and T. Huang, "An Improved Absolute-Cosine Chaotification Model and Its Simple Application in PRNG," in IEEE Access, vol. 11, pp. 59346-59356, 2023.
[4]	I. Syafalni, G. Jonatan, N. Sutisna, R. Mulyawan and T. Adiono, "Efficient Homomorphic Encryption Accelerator with Integrated PRNG Using Low-Cost FPGA," in IEEE Access, vol. 10, pp. 7753-7771, 2022.
[5]	S. Sony, A. P. V and G. R.S., "Design and Analysis of Multi-Bit Linear Feedback Shift Register based PRNG with FPGA Implementation using different Primitive Polynomials," 2022 2nd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2022, pp. 1-7.
[6]	S. Dahmani, M. Maamoun, G. Zerari, N. Chabini and R. Beguenane, "An Efficient FPGA-Based Gaussian Random Number Generator Using an Accurate Segmented Box–Muller Method," in IEEE Access, vol. 11, pp. 64745-64757, 2023.
[7]	S. Akter, K. Khalil and M. Bayoumi, "Efficient Pseudo Random Number Generator (PRNG) Design on FPGA," 2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS), Richardson, TX, USA, 2024, pp. 1-5.
[8]	V. Singh and K. P. Sarawadekar, "FPGA Implementation of Chaos based Pseudo Random Number Generator," 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Dehradun, India, 2021, pp. 1-6.
[9]	R. Amdouni, A. Bradai and M. A. Hajjaji, "FPGA implementation of Chaotic pseudo-random number generator for cryptographic applications," 2024 IEEE 7th International Conference on Advanced Technologies, Signal and Image Processing (ATSIP), Sousse, Tunisia, 2024, pp. 512-517.
[10]	A. Marghescu, P. Svasta, and E. Simion, “High speed and secure variable probability pseudo/true random number generator using FPGA,” in IEEE 21st

	International Symposium on Design and Technology in Electronic Packaging, pp. 323–328, 2015.
[11]	L. G. de la Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, and C. Mancillas-López, “Hardware implementation of pseudo-random number generators based on chaotic maps,” Nonlinear Dynamics, Vol. 90, pp. 1661–1670, 2017.
[12]	L. M. Barakat, A. S. Mansingka, A. G. Radwan, “Hardware stream cipher with controllable chaos generator for colour image encryption,” IET Image Processing, Vol. 8, No. 1, pp. 33–43, 2014.
[13]	Y. Liu and X. Tong, “Hyperchaotic system-based pseudorandom number generator,” IET Information Security, Vol. 10, No. 6, pp. 433–441, 2016.
[14]	C. Wannaboon and P. Ketthong, "A Simple Random-Bit Generator Implemented on FPGA Based on Signum Chaotic Map," 2022 International Conference on Digital Government Technology and Innovation (DGTi-CON), Bangkok, Thailand, 2022, pp. 101-104.
[15]	A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, “A PUF-enabled secure architecture for FPGA-based IoT applications,” IEEE Transaction on Multi-Scale Computing System, Vol. 1, No. 2, pp. 110 – 112, 2015.
[16]	X. Xu and Y. Wang, “High speed true random number generator based on FPGA,” in International Conference on Information Systems Engineering (ICISE), pp. 18-21, 2016.
[17]	F. Farabi F, M. R. Mosavi, and S. Karami, “Optimal choice of random variables in D-ITG traffic generating tool using evolutionary algorithms,” Iranian Journal of Electrical and Electronics Engineering, Vol. 11, No. 2, pp. 101 – 108, 2015.
[18]	A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” National Institute of Standards & Technology (NIST), Apr. 2010, [Online]. Available:https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf
[19]	M. A. Şarkışla and S. Ergün, “An area efficient true random number generator based on modified ring oscillators,” in IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Chengdu, pp. 274 – 278, 2018.
[20]	S. V. Suresh and W. P. Burleson, “Entropy and energy bounds for metastability based TRNG with lightweight post-processing,” IEEE Transaction on Circuits and

	Systems I, Vol. 62, No. 7, pp. 1785 – 1793, 2015.
[21]	H. Martin, P. Peris-Lopez, J. E. Tapiador, and E. San Millan "A new TRNG based on coherent sampling with self-timed rings," IEEE Transactions on Industrial Informatics, Vol. 12, No. 1, pp. 91– 100, 2016.
[22]	Jeong, Y.-S., Oh, K., Cho, C.-K., et al. "Pseudo random number generation using LSTMs and irrational numbers", IEEE International Conference on Big Data and Smart Computing, Shanghai, China, pp. 541–544, 2018.
[23]	D. Lubicz and N. Bochard, "Towards an oscillator based TRNG with a certified entropy rate," IEEE Transaction on Computers, vol. 64, no. 4, pp. 1191–1200, 2015.
[24]	R. Stewart, B. Leung, and G. Gong, "Truly random number generator based on ring oscillator utilizing last passage time," IEEE Transaction on Circuits System II, Express Briefs, vol. 61, no. 12, pp. 937–941, 2014.
[25]	Wu, X.J., Bai, C.X., Kan, H.B, "A new color image cryptosystem via hyperchaos synchronization", Communication Nonlinear Science Number. Simulation, 19, (6), pp. 1884–1897, 2014.
[26]	J. J. R. Panduro, E. C. B. Alvarez, M. d. L. R. Becerra, C. A. O. Avila, M. L. B. Valdovinos and S. Ortega-Cisneros, "Design and Implementation in FPGA a Random Number Generator of 10 bits validated by NIST Maurer's "Universal Statistical" and Binary Matrix Rank tests," 2022 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE), Cuernavaca, Mexico, 2022, pp. 158-163.
[27]	Remya Justina, Binu K Mathew and Susan Abe, "FPGA Implementation of High-Quality Random Number Generator using LUT based Shift Registers," in Elsevier Procedia Technology, vol. 24, pp 1155 – 1162, 2016.
[28]	D. B. Thomas and W. Luk, "The LUT-SR Family of Uniform Random Number Generators for FPGA Architectures", IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol. 21, No. 4, pp. 761 – 770, April 2013.
[29]	O. Cret, A. Suciu, and T. Gyorfi, "Practical issues in implementing TRNGS in FPGAS based on the ring oscillator sampling method," in Proceedings of 10th International Symposium on Symbol Number Algorithms Science Computing (SYNASC'08), pp. 433–438, 2008.
[30]	Andrei Marghescu, Paul Svasta and Emil Simion, "High Speed and Secure Variable Probability Pseudo/True Random Number Generator using FPGA," in IEEE 21st International Symposium on Design and Technology in Electronic Packaging, 2015.

[31]	A. P. Johnson, R. S. Chakraborty, and D. Mukhopadyay, “An improved DCM-based tunable true random number generator for Xilinx FPGA,” IEEE Transactions on Circuits and Systems—II: Express Briefs, Vol. 64, No. 4, pp. 452–456, 2017.
[32]	Dongsheng Liu, Zilong Liu, Lun Li, and Xuecheng Zou, “A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards”, IEEE Transactions on Circuits and Systems-II: Express Briefs, Vol. 63, No. 6, pp. 608 - 612, 2016.
[33]	Miguel Garcia-Bosque, Adrián Pérez-Resa, Carlos Sánchez-Azqueta, Concepción Aldea and Santiago Celma, “Chaos-Based Bitwise Dynamical Pseudorandom Number Generator on FPGA,” IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 1, pp. 291 – 293, 2019.
[34]	Seongmo Park, Byoung Gun Cho, Taewook Kang, Kyunghwan Park, Youngsu Kwon and Jongbum Kim, “Efficient hardware implementation and analysis of true random number generator based on beta source,” ETRI Journal, vol. 42, no. 4, pp. 518–526, 2020.
[35]	Yiming Wen and Weize Yu, “Machine learning-resistant pseudo-random number generator,” IEEE Electronics Letters, vol. 55, No. 9, pp. 515–517, 2019.
[36]	Ghada Elsayed, Elsayed Soleit and Somaya Kayed, “FPGA design and implementation for adaptive digital chaotic key generator,” Springer Bulletin of the National Research Centre, vol. 47, no. 122, pp. 1 – 9, 2023.
[37]	T. L. Liao, P. Y. Wan1, Jun-Juh Yan, “Design and Synchronization of Chaos-Based True Random Number Generators and Its FPGA Implementation,” IEEE Access, vol. 10, pp. 8279 – 8286, 2022.